Chapter 4

High-Level Recognition

A high-level recognizer transforms graphical symbols recognized by the low-level recognizer into commands for creating and manipulating diagrams. In contrast to the low-level recognition, the high-level recognition is editor-dedicated and language-oriented. Therefore, we first introduce the diagram class to which our high-level recognizer is designed, and provide a few formal definitions for handsketch-based editing. Then we concentrate on the fundamental concepts of the high-level recognition, and describe the system components with a number of representative examples.

4.1 Formal Basis

In this section, we formally define the syntactical elements of a very frequently used diagram class by describing the visual alphabets and the visual compositions. We establish an appropriate graph model for the internal representation, and consider both representation forms simultaneously in the definition of diagram schema. Then, we introduce the concept of gesture operator and handsketch-based editing.

4.1.1 HiNet Diagrams

In general, diagrams refer to all two-dimensional representations of any kind of information. There are countless diagrams which are used in everyday’s life and in science. For example, diagrams for division or multiplication, coordinate dia-
grams for mathematical functions, and "cake"-diagrams for comparing percentages as shown in figure 4.1.

\[
\begin{array}{c}
365 : 7 = 52.1 \\
35 \\
15 \\
14 \\
10 \\
7 \\
3
\end{array}
\]

Figure 4.1: Examples of general diagrams

The meaning of diagrams in this work is more specific than this, they are considered as visual programs of certain diagram languages. Diagram languages are the most frequently used visual languages in computer aided design, software engineering, or structural analysis. They play an essential part in designing complex systems and developing programs. Several decades ago, flowcharts were used to design and illustrate programs. Nowadays we have SADT [78], Nassi-Shneiderman diagrams [85], state transition diagrams, statecharts [43], Petri nets [98], Express [114], SpecChart [126], entity relationship charts, semantic nets, influence diagrams [123] and many others. Figure 4.2 illustrate several different diagrams from this category.

The enthusiasm of development and application of so many diagram languages due to the following significant advantages of diagram languages:

- Diagrams give an aid to clear thinking and enforce good structuring. Relationships between objects can be represented much clearly in a diagram than in a textual program.

- Diagrams allow quick and precise communication between members of a development team, because human perception is naturally two or three dimensional rather than one dimensional [104].

These diagram languages are widely used for purposes like analysis, specification, design, modeling, or description of various structures. The syntactic structures of